Hotplug of Non-discoverable Hardware: Status and Future Directions

Luca Ceresoli
luca.ceresoli@bootlin.com

Embedded Linux Conference Europe 2025

© Copyright 2004-2025, Bootlin.
Creative Commons BY-SA 3.0 license.
Corrections, suggestions, contributions and translations are welcome!

Luca Ceresoli

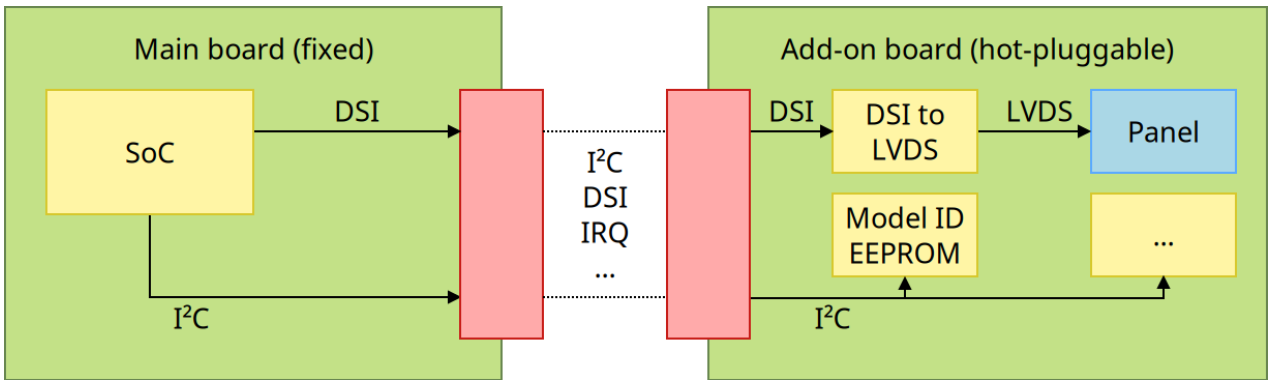
- Embedded Linux engineer at **Bootlin**
  - Development, consulting and training about **embedded Linux**
  - Open-source focus
- **Linux kernel** device driver developer
- Bootloaders, Buildroot and Yocto integration
- Open-source contributor
- Living in **Bergamo**, Italy

Agenda

- Use case and goals
- The connector abstraction
- Connector: current status
- DRM: current status

Use case and goals

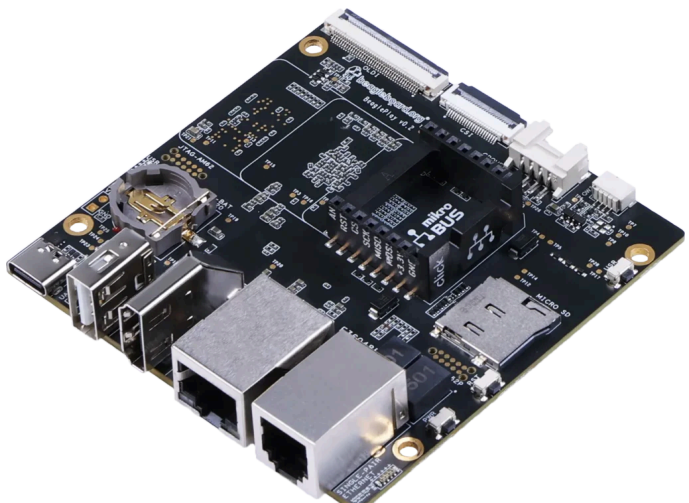
Embedded product with add-ons



- A fairly normal ARM64 embedded system, using device tree...
- ...except it has a connector for an add-on to extend features
  - Proprietary connector
  - Can be inserted and removed by user at any moment
  - Connector uses non-discoverable resources: GPIO, interrupts, I2C, MIPI DSI...
  - Multiple add-on models supported
    - Add-ons have an EEPROM with model ID

Developer boards with expansion connectors

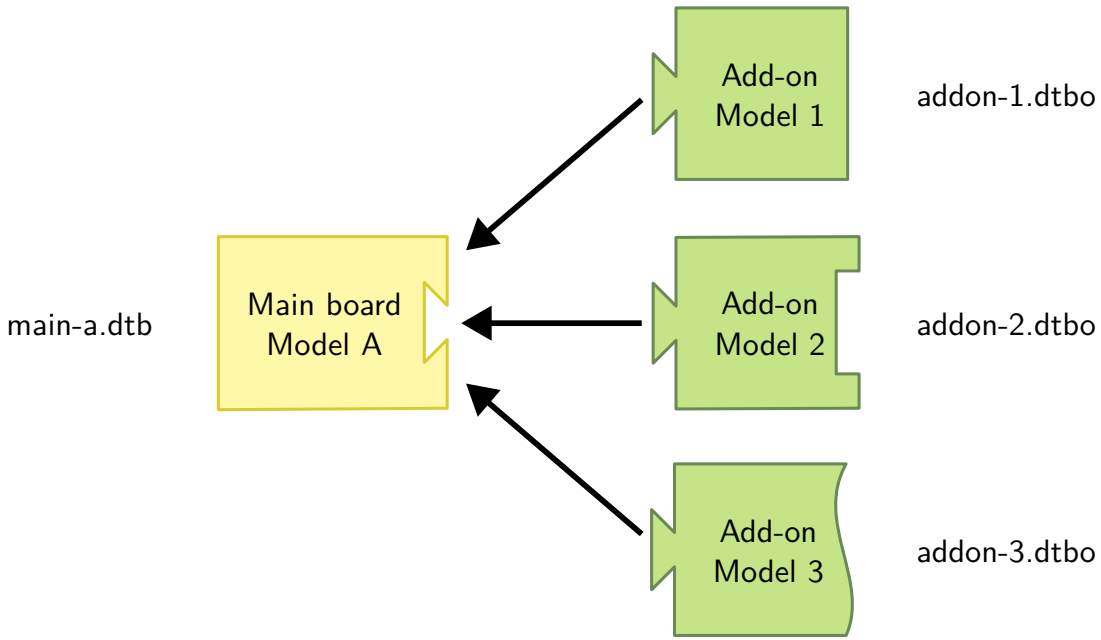
- BeaglePlay has a connector for add-ons
- Not hot-pluggable, but otherwise has similar requirements
- Ayush Singh is working on this use case



Hotplug and device tree

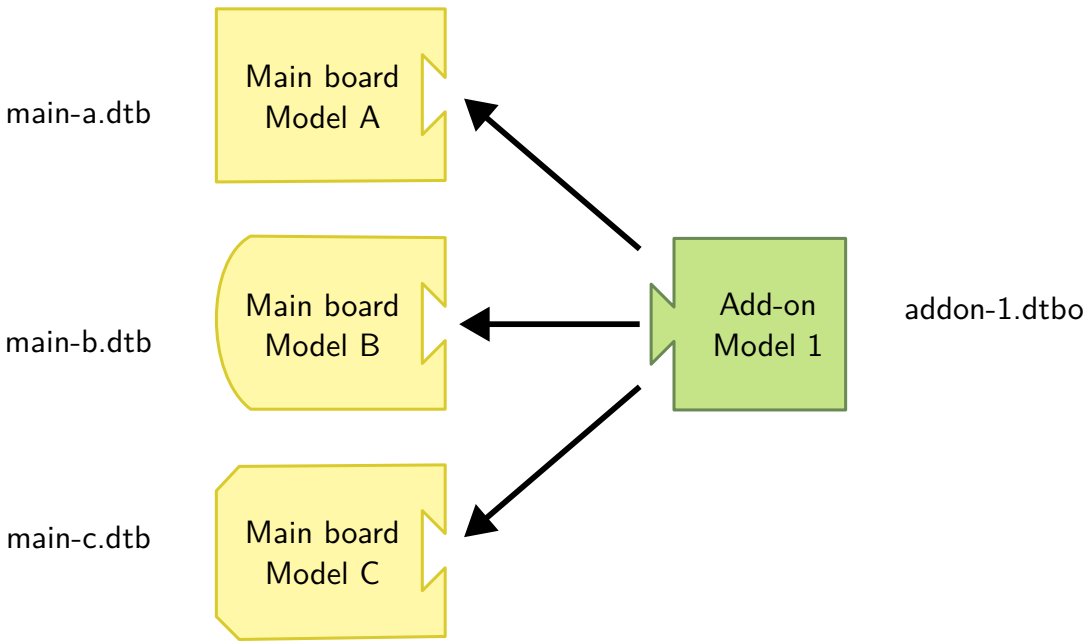
- Main board has a classic device tree (.dts/.dtb)
- Addon has a **device tree overlay** (.dtso/.dtbo)
  - Looks like the natural tool to describe added hardware
- There is a **connector driver**
  - Detects hot-plug → loads overlay → add-on hardware populates
  - Detects hot-unplug → removes overlay → add-on hardware depopulates

Connectors allow decoupling 1/3



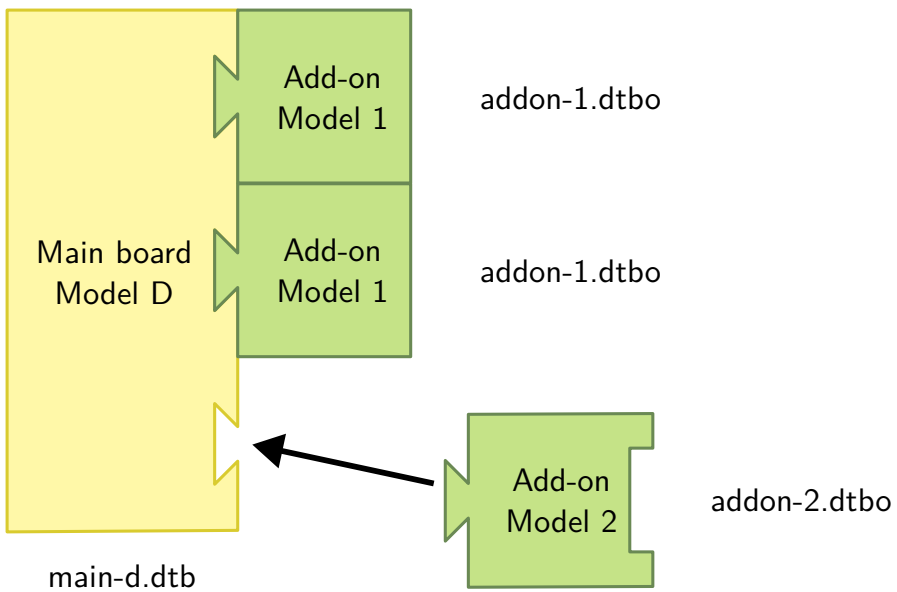
- Any addon model can be connected to the “host”

Connectors allow decoupling 2/3



- An addon can be connected to any main board model

Connectors allow decoupling 3/3

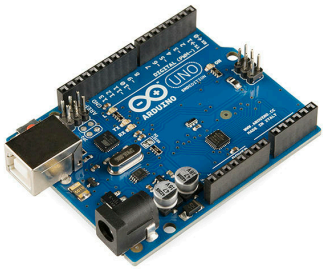


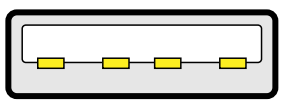
- A mainboard can have multiple connectors...
- ...all can be connected at any time
  - even with the same addon model → same DT overlay

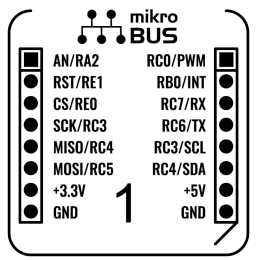
The **connector** abstraction

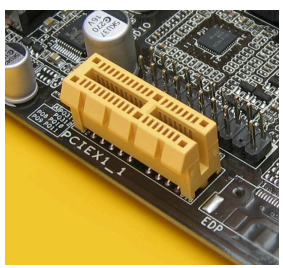
Generalization: a connector

- Connector = a well-defined standard = a contract
  - Mechanical definition
  - Electrical (voltages, max current, impedance, ...)
  - Pinout (which function for each pin)
  - Protocol to read model ID from addon
    - If multiple models need to be handled differently
    - Optional (model can be known "by other means")
- Not all connectors support hotplug









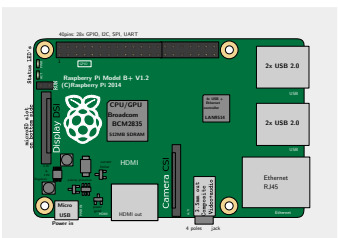
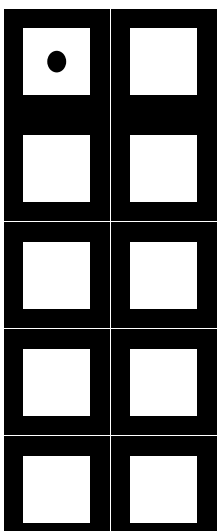


Image references:
https://commons.wikimedia.org/wiki/File:Schuko_plug_and_socket.png
https://commons.wikimedia.org/wiki/File:USB_Type-A_receptacle_White.svg
https://en.wikipedia.org/wiki/PCI_Express#/media/File:PCIe_J1900_SoC_ITX_Mainboard_IMG_1820.JPG https://commons.wikimedia.org/wiki/File:Arduino_Uno_-_R3.jpg
https://www.mikroe.com/mikrobus
https://en.wikipedia.org/wiki/Raspberry_Pi#/media/File:Raspberry_Pi_B+_rev_1.2.svg

Connector specification

- Defines *function* of each pin:
  - Provided by the main board
  - Consumed by the add-on

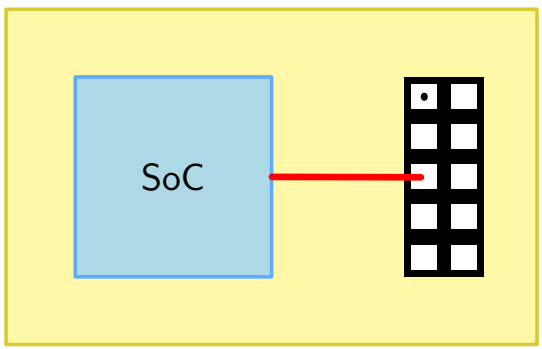


Basic example:

- Pin 1: I2C1 SCL
- Pin 2: I2C1 SDA
- Pin 3: I2C2 SCL
- Pin 4: I2C2 SDA
- Pin 5: GPIO 1
- Pin 6: 5 Vdc 1 A
- Pin 7: GPIO 2
- Pin 8: 15 Vdc 100 mA
- Pin 9: GND
- Pin 10: GND

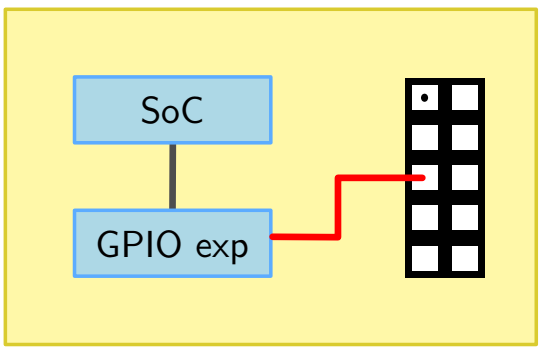
Provider specification

- Main board device tree describes how pin functionality is provided
  - Components used to provide it



main-a.dtb

- GPIO1 (Pin 5) → `<&soc_gpio3 12 0>`

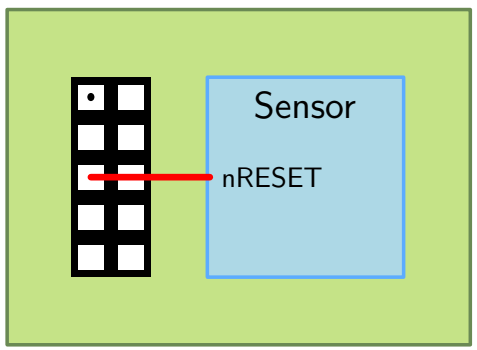


main-b.dtb

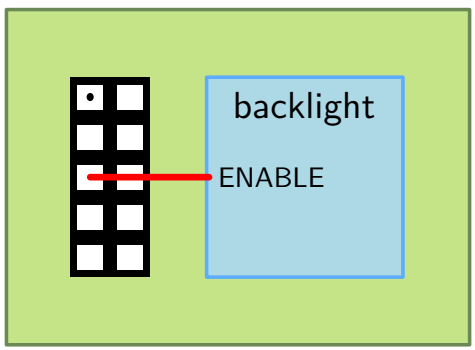
- GPIO1 (Pin 5) → `<&tca6424 19 0>`

Consumer specification

- Addon device tree overlay describes how the connector pins are used
  - Components comsuming it



addon-1.dtbo



addon-2.dtbo

- `reset-gpios = <&connector 1 ACTIVE_LOW>`
  → GPIO1 (Pin 5)
- `enable-gpios = <&connector 1 ACTIVE_HIGH>`
  → GPIO1 (Pin 5)

Connector: current status

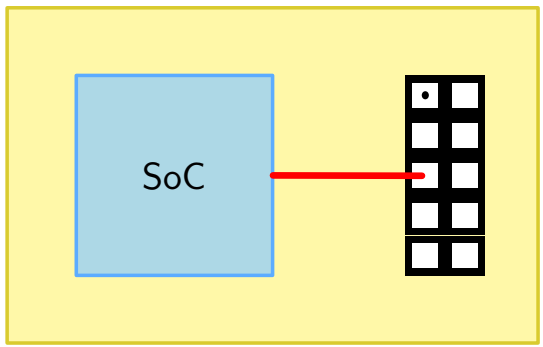
Preliminary work (historical)

- Initial proposal
  - 2024-05-10: `[PATCH v2 0/5] Add support for GE SUNH hot-pluggable connector`
    - first with connector dt-bindings and a connector driver
  - 2025-02-06: `[PATCH v6 00/26] Add support for hot-pluggable DRM bridges`
    - Most recent complete draft
    - After that, sent separate series for various aspects
- Discussion at Linux Plumbers Conference 2024
  - *Runtime hotplug on non-discoverable busses with device tree overlays*
  - by Luca Ceresoli and Hervé Codina
  - `https://lpc.events/event/18/contributions/1696/`
  - Very good discussion, led to many improvements, approach became mature

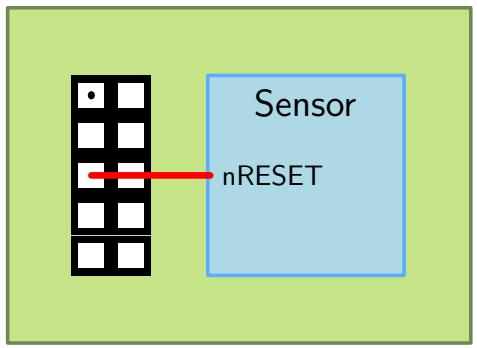
Connector device tree bindings: `export-symbols` 1/5

- Proposal to add new `export-symbols` in device tree
  - Inspired to `__symbols__`
  - But overlay cannot reference nodes beyond the connector
- Allow full decoupling the main board (producer) and addon (consumer)
- Using Nexus nodes for GPIOs, interrupts and PWM

Connector device tree bindings: export-symbols 2/5



main-a.dtb



addon-1.dtbo

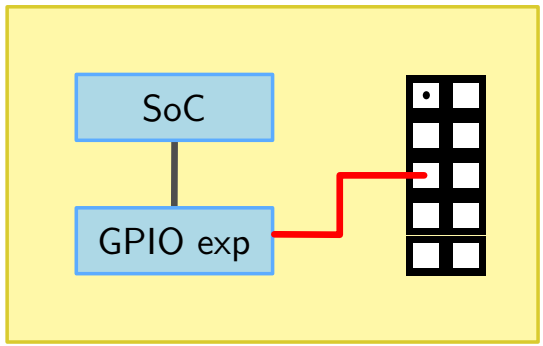
Main board A DT:

```
front_connector: addon-connector {
    #gpio-cells = <2>;
    gpio-map = <1 0 &soc_gpio3 12 0>;
    export-symbols {
       connector = <&front_connector>;
    };
};
```

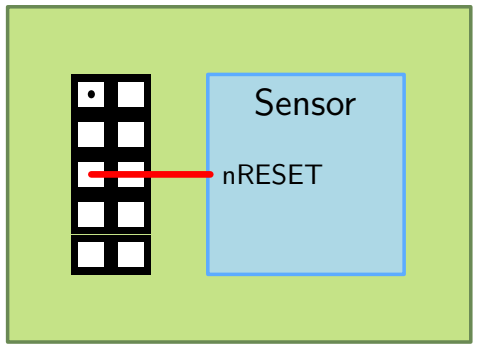
Addon DT overlay:

```
node {
// reset-gpios = <&soc_gpio3 12 GPIO_ACTIVE_HIGH>; NO!
   reset-gpios = <&connector  1 GPIO_ACTIVE_HIGH>;
};
```

Connector device tree bindings: export-symbols 3/5



main-b.dtb



addon-1.dtbo

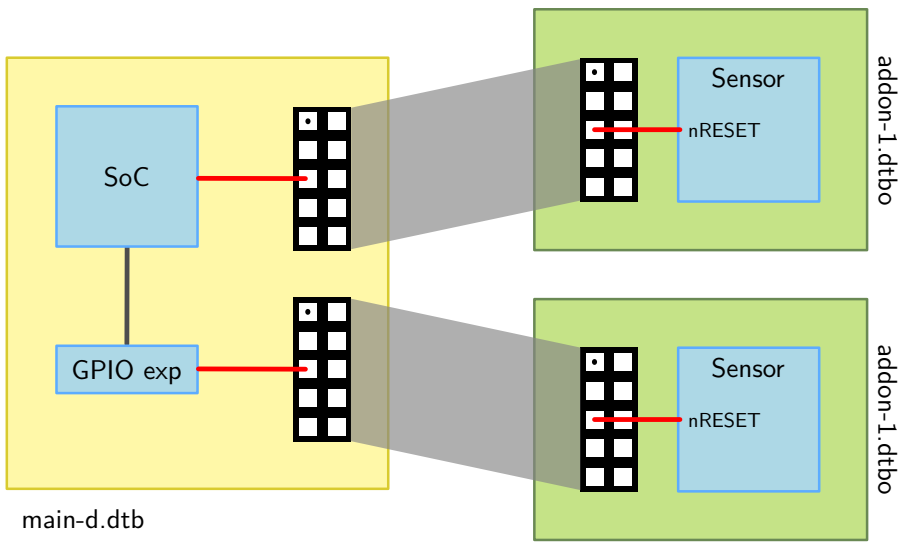
Main board B DT:

```
side_connector: addon-connector {
    #gpio-cells = <2>;
    gpio-map = <1 0 &tca6424 19 0>;
    export-symbols {
       connector = <&side_connector>;
    };
};
```

Addon DT overlay:

```
node {
// reset-gpios = <&tca6424   19 GPIO_ACTIVE_HIGH>; NO!
   reset-gpios = <&connector  1 GPIO_ACTIVE_HIGH>;
};
```

Connector device tree bindings: export-symbols 4/5



main-d.dtb:

```
connector1: addon-connector1 {
    gpio-map = <1 0 &soc_gpio2 42 0>;
    export-symbols {
       connector = <&front_connector>;
    };
};
connector2: addon-connector2 {
    gpio-map = <1 0 &tca6424 11 0>;
    export-symbols {
       connector = <&front_connector>;
    };
};
```

addon-1.dtbo (used twice):

```
reset-gpios =
    <&connector 1 GPIO_ACTIVE_HIGH>;
```

Connector device tree bindings: `export-symbols` 5/5

- Proposal sent by Hervé Codina
  - Bindings and implementation in the Linux kernel
  - `of: overlay: Add support for export-symbols node feature`
  - `2024-12-09: v1`
  - `2025-04-30: v2`
- Proposal sent by Ayush Singh
  - Proposing the same, but to DT spec and with implementation in `dtc` and `fdtoverlay`
  - Targeting developer boards (BeaglePlay)
  - 2025-04-11: `[PATCH v3] Add new export-symbols node` (DT spec)
  - 2025-01-10: `[PATCH] checks: Add support for export-symbols` (dtc)
  - 2025-01-11: `[PATCH] libfdt: overlay: Add export-symbols support` (fdtoverlay)
- Some feedback recently
  - Krzysztof Kozlowski raised syntax issues
  - Rob Herring skeptical about the idea, but Hervé found counter-proposals limited

I²C bus extension

- Devices are subnodes of bus
- Nexus nodes not suitable

I²C bus extension: current proposal

Main board:

```
soc_i2c1: i2c@abcd0000 {
    compatible = "xyz,i2c-ctrl";
};
soc_i2c5: i2c@cafe0000 {
    compatible = "xyz,i2c-ctrl";
};
connector {
    i2c_ctrl: i2c-ctrl {
        compatible = "i2c-bus-extension"; //??
        i2c-parent = <&soc_i2c1>;
        #address-cells = <1>;
        #size-cells = <0>;
    };
    i2c-sensors {
        compatible = "i2c-bus-extension"; //??
        i2c-parent = <&soc_i2c5>;
        #address-cells = <1>;
        #size-cells = <0>;
};  };
```

Overlay:

```
i2c-ctrl {
    eeprom@50 {
        compatible = "atmel,24c64";
    };
};
```

← `//??` = under discussion

I²C bus extension: discussion

- Linux Plumbers Conference 2024
  - Proposal made but incomplete
  - Discussed with Wolfram Sang (I²C maintainer)
    - Led to bidirectional links, recently rejected by Rob Herring, removed
- Latest proposal
  - Bindings to dt-schema (where I²C is specified)
    - 2025-04-30: `[PATCH v2 0/1] i2c: Introduce I2C bus extensions`
  - Kernel implementation
    - 2025-02-05: `[RFC PATCH 0/3] i2c: Introduce i2c bus extensions`
    - Good feedback from Wolfram but on hold, waiting for bindings
- 2025-08-01: discussion restarted, now ongoing

SPI bus extension

- Ayush proposed for SPI the same approach as I²C
  - 2025-07-29: `[PATCH 0/4] spi: Introduce spi bus extensions`
  - Bindings and implementation in the Linux kernel
  - Not much discussion: the outcome for I²C will apply to SPI too

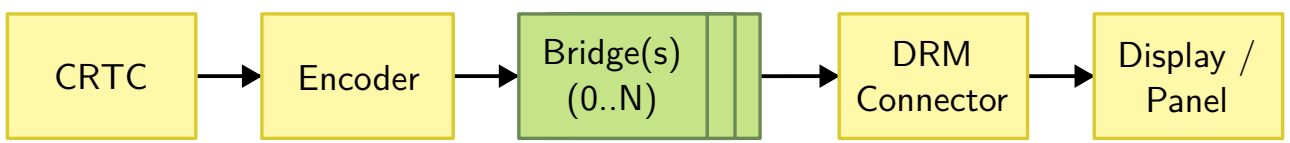
A different approach: global VS local DT overlays

- Local
  - As discussed so far: overlay does not reference anything beyond the connector
  - Based on `export-symbols`
- Global
  - Proposed by Andrew Davis
  - Based on the old `__symbols__` node
    - Overlay can touch any node in the base tree, long-time rejected
  - Uses adapter dtbo + macros to decouple mainboard and addon
  - Does not support removal
- 2025-04-30: `[Discussion] Global vs Local devicetree overlays for addon board + connector setups`
  - Started by Ayush

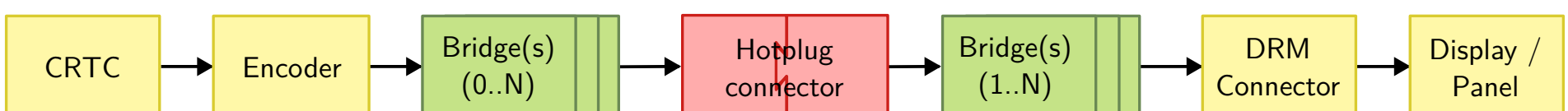
DRM:
current status

The problem

- The problem
  - DRM supports hot-plugging displays after a connector, not other components
  - The hot-plug add-on has a bridge
  - Pointers to `struct bridge` are taken by various parts of the DRM stack



A classic DRM pipeline



A DRM pipeline with hot-pluggable bridges

Preliminary work (historical)

- Initial proposal
  - 2024-03-26: `[PATCH 0/4] drm: add support for hot-pluggable bridges`
    - Based on a hotplug-bridge, decoupling the two sides
  - 2025-02-06: `[PATCH v6 00/26] Add support for hot-pluggable DRM bridges`
    - Most recent complete draft
    - After that, sent separate series for various aspects
- Discussion at Linux Plumbers Conference 2024
  - *Hotplug DRM pipeline components on non-discoverable video busses*
  - by Luca Ceresoli
  - `https://lpc.events/event/18/contributions/1750/`
  - A lot of discussion, largely in the hallway → new Grand Plan
- Grand Plan progressing
  - Lots of discussion, mostly with Maxime Ripard

The Grand Plan

1. Introduce dynamic DRM bridge lifetime (`struct drm_bridge`)
2. Handle gracefully atomic updates during bridge removal
3. DSI host-device driver interaction
4. Finish the hotplug bridge work

1. Introduce dynamic DRM bridge lifetime

- Problem:
  - Until 6.15: bridge drivers allocate and free the bridge (usually `devm_kzalloc()`)
  - Other drivers can take a pointer to the bridge
    - No problem: the whole card is removed at once... until now
  - When bridge removed → use-after-free 💣
- Solution: dynamic bridge lifetime
  - Add refcount to `struct drm_bridge`
  - Add `devm_drm_bridge_alloc()` (mandatory) to allocate bridge and initialize refcount
  - Any code taking a pointer must `drm_bridge_get/put()` it
  - Memory is freed at the last put
    - Possibly long after hardware *and* driver removed

1. Introduce dynamic DRM bridge lifetime 1/4

A. Add new `devm_drm_bridge_alloc()` API and refcounting (✓ in v6.16)
B. Convert all bridge drivers to new API (✓ in 6.17-rc1)
   - Most conversions are trivial (see `examples`)
C. `Kunit tests` (✓ in 6.17-rc1)
D. Add get/put to `drm_bridge_add/remove()` + `drm_bridge_attach/detach()`, warn on old allocation pattern (✓ in 6.17-rc1)

1. Introduce dynamic DRM bridge lifetime 2/4

E. Add get/put on bridge users
   1. `Add a cleanup action, drm_bridge_chain_get_first_bridge()` (✓ in drm-misc-next → 6.18)
   2. `drm_bridge_get_prev_bridge()` (✓ in drm-misc-next → 6.18)
   3. `drm_bridge_get_next_bridge()` (✓ fully reviewed → likely 6.18)
   4. `drm_for_each_bridge_in_chain()` (under review)
   5. `drm_bridge_connector_init()` (on hold, depends on items 3 and 4)
   6. `of_drm_find_bridge()`
      - Problem: used by functions in item 7
      - Plan: `add drm_of_find_bridge(), then deprecate`
   7. `drm_of_find_panel_or_bridge()`, `*_of_get_bridge()`
      - Complex due to broken semantics
      - The `DRM panel rework` by Anusha Srivatsa (in progress) might help

1. Introduce dynamic DRM bridge lifetime 3/4

F. Debugfs improvements

- Show all bridges in `/sys/kernel/debug/dri/bridges`, show refcount (✓ in 6.16)

```
# cat /sys/kernel/debug/dri/bridges
bridge[0]: hotplug_bridge_driver_exit [hotplug_bridge]
        type: [16] DSI
        OF: /addon-connector/dsi:
        ops: [0x5] detect hpd
bridge[1]: samsung_dsim_driver_exit [samsung_dsim]
        type: [16] DSI
        OF: /soc@0/bus@32c00000/dsi@32e60000:fsl,imx8mp-mipi-dsim
        ops: [0x0]
...
#
```

1. Introduce dynamic DRM bridge lifetime 4/4

F. Debugfs improvements

- Show bridges removed but not yet freed, show refcount (under review)

```
# cat /sys/kernel/debug/dri/bridges
bridge[0]: hotplug_bridge_driver_exit [hotplug_bridge]
        addr: ffff000001d0e008
        refcount: 5
        type: [16] DSI
        OF: /addon-connector/dsi:
        ops: [0x5] detect hpd
bridge[1]: samsung_dsim_driver_exit [samsung_dsim]
        addr: ffff000002d80420
        refcount: 2 [removed]
        type: [16] DSI
        ops: [0x0]
...
#
```

Intermezzo: a new warning

- If you see this...

```
[drm] DRM bridge corrupted or not allocated by devm_drm_bridge_alloc()
------------[ cut here ]------------
refcount_t: addition on 0; use-after-free.
WARNING: CPU: 2 PID: 83 at lib/refcount.c:25 refcount_warn_saturate+0x120/0x148
[...]
Call trace:
 refcount_warn_saturate+0x120/0x148 (P)
 drm_bridge_get.part.0+0x70/0x98 [drm]
 drm_bridge_add+0x34/0x108 [drm]
 sn65dsi83_probe+0x200/0x480 [ti_sn65dsi83]
 [...]
```

- ...then perhaps a driver was not yet converted to `devm_drm_bridge_alloc()`
- The first line was added in 6.17-rc1 to make the core reason easy to find

2. Handle gracefully atomic updates during bridge removal

- Problem:
  - There are multiple *concurrent* entry points to a DRM bridge driver code
    - Atomic callbacks
    - New: driver `.remove` on hot-unplug
  - Device resources could be accessed while being freed
- Solution: protect access to device resources
  - Access to device resources guarded by `drm_bridge_enter()` and `drm_bridge_exit()`
  - `.remove` calls `drm_bridge_unplug()` to declare "bridge was unplugged"
  - Based on `drm_dev_unplug/enter/exit()` for the DRM device
- First simple attempt:
  - `drm/bridge: handle gracefully atomic updates during bridge removal`
  - under review

3. DSI host-device driver interaction 1/2

- Problem: Most DSI host drivers do `drm_bridge_add()` in the DSI attach callback, not in probe
  - Feels unnatural to me 🤷
    - Really wrong? Under discussion
  - DRM card won't populate before DSI host bridge adds itself
    - DRM card not available when booting without addon
    - Needs a horrible workaround in the hotplug-bridge (see `the hotplug_bridge_dsi_attach() documentation`)
  - DSI device hot(un)plugs multiple times → multiple `drm_bridge_add/remove()`
    - Complicates debugfs code to show "re-added" bridges (see above)

3. DSI host-device driver interaction 2/2

- Proposed solution: move `drm_bridge_add()` to probe (tested for one driver)
- `samsung-dsim: move drm_bridge_add() call to probe`
- Root issues under discussion

4. Finish the hotplug bridge work

- Direction depends on previous steps
- Need to remove the “always-connected” LVDS connector
- Keep the `hotplug-bridge` or remove it?
  - Removal would leave “some burden” on all bridges that support hotplugging the following bridge, to be evaluted
- Possibly other changes needed

Conclusions

Conclusions

- Device tree
  - Still blocked at device tree specification
    - Unconference session for discussion today at 16:20
  - Implementation well tested for the current DT spec proposal
- DRM
  - There's a plan, 4 chapters
  - Chapter 1 progressing quite smoothly
  - Chapters 2 and 3 slowly moving out of the fog
  - Chapter 4 still largely obscure

Thank you!

Questions?

Luca Ceresoli

luca.ceresoli@bootlin.com

Slides under CC-BY-SA 3.0

https://bootlin.com/pub/conferences/